Cnuas Control Plane, Datasheet

Unified control plane, one command line and one REST API over every component

Document DS-CNU-020 • **Revision** A • **Issued** 03 August 2026 • **Status** Published

Item	Value
Part	`cnuas`, `cnuas-api`
Type	Control plane, command line and REST service
Package	`cnuas`, version 0.1.0
Version	`9fbdddf-dirty`
Repo	PacketFive/cnuas

1. Overview

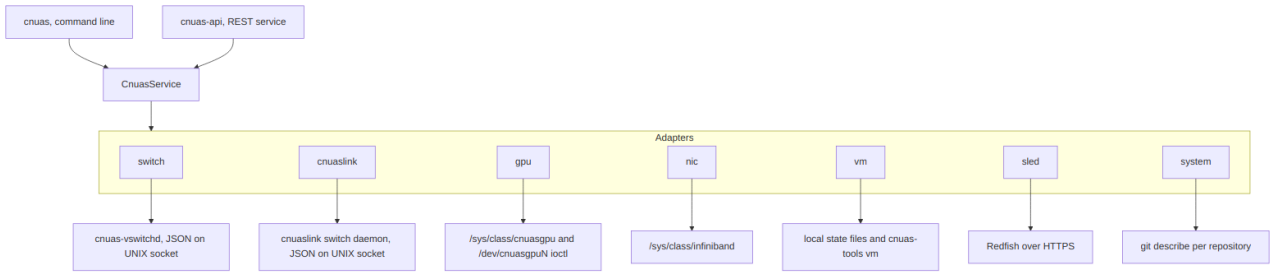
The Cnuas control plane is the single operator surface for a virtual data centre. It presents one command line client and one REST service over every component in the platform: the fabric switch, the accelerator link switch, the accelerators, the RDMA adapters, the guest virtual machines, and the sled management controllers.

Both front ends call the same service object, so anything scriptable from the command line is available over HTTP with identical semantics. Components are reached through adapters, one per component, each of which owns the transport detail for its target.

Key features

- One command tree over seven component families.
- REST service with generated OpenAPI description and interactive documentation.
- Machine readable output from every command with a single flag.
- Health probe that reports reachability of each component independently.
- Version reporting resolved from the git tags of each component repository.
- Adapters isolate transport, so a component can change protocol without changing the operator surface.

2. Architecture



3. Functional specifications

Parameter	Value
Package version	0.1.0
Python	3.10 or later
Command line entry point	`cnuas`, built on Typer
REST entry point	`cnuas-api`, built on FastAPI
Command groups	`system`, `switch`, `fabric`, `gpu`, `nic`, `vm`, `sled`
Commands	30 plus `version` and `api`
REST routes	39 operations across 35 paths
API prefix	`/api/v1`
Interactive documentation	`/docs`
Machine description	`/openapi.json`
Output modes	Rich tables by default, JSON on request
Authentication	None in this revision, see section 8

4. Command line reference

Group	Commands
`system`	`health`, `inventory`, `versions`
`switch`	`ports`, `set-mode`, `set-link`, `set-pfc`, `set-ecn`, `set-ets`, `fdb`, `lft`, `sm`, `telemetry`
`fabric`	`ports`, `gpu-fdb`, `set-link`, `telemetry`, `version`
`gpu`	`list`, `info`, `link`
`nic`	`list`, `info`
`vm`	`list`, `status`, `up`, `down`, `reset`, `lab`
`sled`	`list`, `power`
root	`version`, `api`

5. REST reference

Area	Routes
Service	`GET /health`, `GET /`
System	`GET /api/v1/system/inventory`, `GET /api/v1/system/versions`
Switch ports	`GET /api/v1/switch/ports`, `GET /api/v1/switch/ports/{port}`, `PUT /api/v1/switch/ports/{port}/mode`, `.../link`, `.../pfc`, `.../ecn`, `.../ets`
Switch forwarding	`GET`, `POST`, `DELETE /api/v1/switch/fdb` and `/api/v1/switch/lft`
Subnet manager	`GET /api/v1/switch/sm`, `POST /api/v1/switch/sm/{action}`
Switch telemetry	`GET /api/v1/switch/telemetry`, `POST /api/v1/switch/telemetry/clear`
Fabric	`GET /api/v1/fabric/ports`, `/gpu-fdb`, `/telemetry`, `/version`, `PUT /api/v1/fabric/link`, `POST /api/v1/fabric/telemetry/clear`
Accelerators	`GET /api/v1/gpu`, `GET /api/v1/gpu/{gpu_id}`, `GET /api/v1/gpu/{gpu_id}/link`
Adapters	`GET /api/v1/nic`, `GET /api/v1/nic/{device}`
Virtual machines	`GET /api/v1/vm`, `GET /api/v1/vm/{name}`, `POST /api/v1/vm/{name}/up`, `.../down`, `.../reset`
Sleds	`GET /api/v1/sled`, `GET /api/v1/sled/{name}/power`

6. Adapters

Adapter	Target	Transport
`SwitchAdapter`	`cnuas-vswitchd`	JSON over UNIX domain socket
`CnuasLinkAdapter`	CnuasLink switch daemon	JSON over UNIX domain socket
`GpuAdapter`	`cnuasgpu.ko`	sysfs class attributes and a device ioctl
`NicAdapter`	`cnuas_ib.ko`	sysfs, `/sys/class/infiniband`
`VmAdapter`	Guest virtual machines	Local state files for reads, `cnuas-tools vm` for changes, Redfish for sled managed blades
`SledAdapter`	Sled management controller	Redfish over HTTPS
`SystemAdapter`	Component repositories	`git describe`

7. Health model

`cnuas system health` probes the switch, the link switch, the accelerators, the adapters and the virtual machines independently and reports each result on its own. A component that is not running is reported as unreachable rather than failing the whole call, so the command is usable as a lab readiness check while components are still coming up.

8. Operating notes

No authentication in this revision

The REST service carries no authentication, authorisation or transport security of its own. It is designed for a laboratory network and for localhost use. Put it behind a reverse proxy that terminates TLS and enforces access control before exposing it beyond a trusted host.

- Read operations are safe to run continuously and are used by the dashboards.
- Mutating virtual machine operations shell out to `cnuas-tools`, so that tool must be installed and on the path for `vm up`, `vm down` and `vm reset`.

9. Validation

Area	Test	Cases
Command line behaviour and output modes	`cnuas/tests/test_cli.py`	7
REST routes and response shapes	`cnuas/tests/test_api.py`	17
Adapter transports and parsing	`cnuas/tests/test_adapters.py`	23
Service composition and health model	`cnuas/tests/test_service.py`	5

All 52 cases pass. See the Validation Matrix.

10. Integration information

Item	Value
Repo	`PacketFive/cnuas`
Source	`cnuas/src/cnuas/`

Item	Value
Entry points	`cnuas`, `cnuas-api`
Runtime dependencies	Typer, Rich, FastAPI
Related	Control Plane design, API reference

11. Revision history

Revision	Notes
A	First publication. Command tree, route list and adapter inventory read from the source.